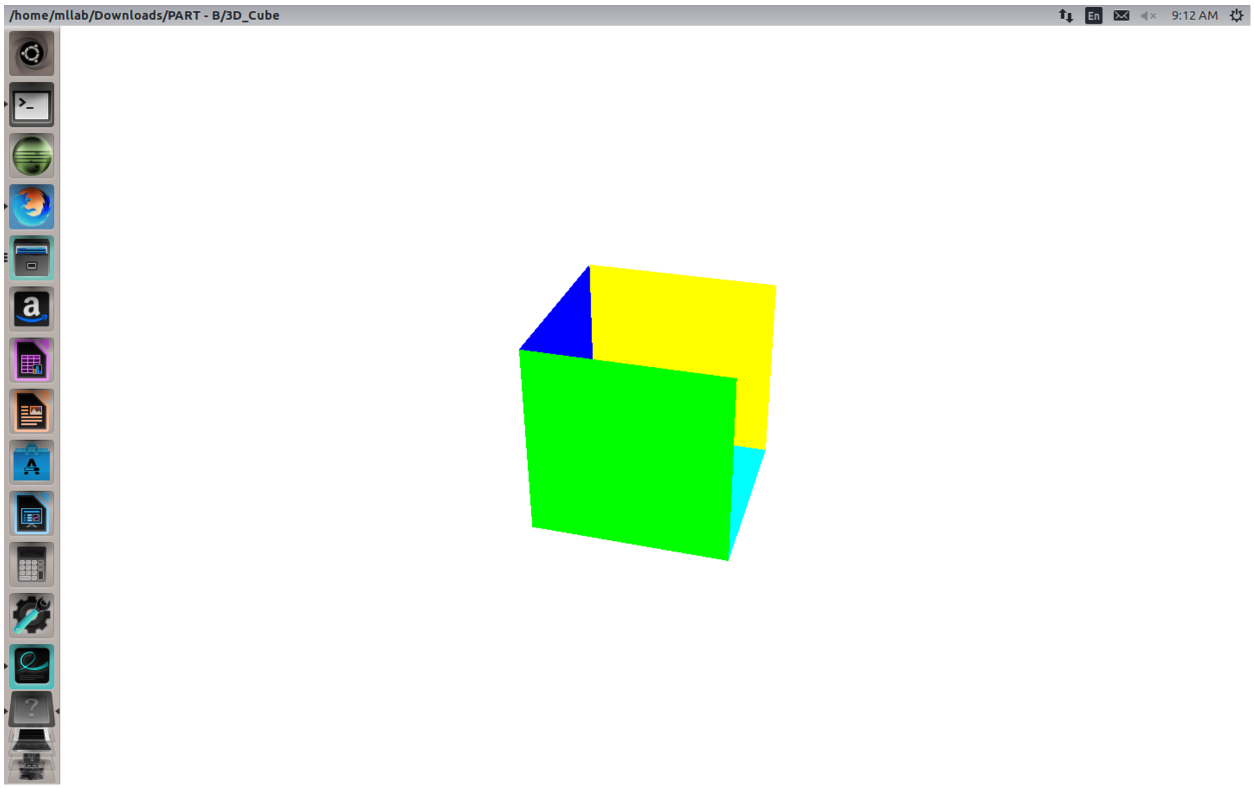
Assignment 8: Animation

```
#include <GL/glut.h>

 GLfloat xRotated, yRotated, zRotated;
void init(void)
{
glClearColor(0,0,0,0);
}

void DrawCube(void)
{

     glMatrixMode(GL_MODELVIEW);
    // clear the drawing buffer.
    glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
        glTranslatef(0.0,0.0,-10.5);
    glRotatef(xRotated,1.0,0.0,0.0);
    // rotation about Y axis
    glRotatef(yRotated,0.0,1.0,0.0);
    // rotation about Z axis
    glRotatef(zRotated,0.0,0.0,1.0);
  glBegin(GL_QUADS);        // Draw The Cube Using quads
    glColor3f(0.0f,1.0f,0.0f);    // Color Blue
    glVertex3f( 1.0f, 1.0f,-1.0f);    // Top Right Of The Quad (Top)
    glVertex3f(-1.0f, 1.0f,-1.0f);    // Top Left Of The Quad (Top)
    glVertex3f(-1.0f, 1.0f, 1.0f);    // Bottom Left Of The Quad (Top)
    glVertex3f( 1.0f, 1.0f, 1.0f);    // Bottom Right Of The Quad (Top)
    glColor3f(1.0f,0.5f,0.0f);    // Color Orange
    glVertex3f( 1.0f,-1.0f, 1.0f);    // Top Right Of The Quad (Bottom)
    glVertex3f(-1.0f,-1.0f, 1.0f);    // Top Left Of The Quad (Bottom)
    glVertex3f(-1.0f,-1.0f,-1.0f);    // Bottom Left Of The Quad (Bottom)
    glVertex3f( 1.0f,-1.0f,-1.0f);    // Bottom Right Of The Quad (Bottom)
    glColor3f(1.0f,0.0f,0.0f);    // Color Red
    glVertex3f( 1.0f, 1.0f, 1.0f);    // Top Right Of The Quad (Front)
    glVertex3f(-1.0f, 1.0f, 1.0f);    // Top Left Of The Quad (Front)
    glVertex3f(-1.0f,-1.0f, 1.0f);    // Bottom Left Of The Quad (Front)
    glVertex3f( 1.0f,-1.0f, 1.0f);    // Bottom Right Of The Quad (Front)
    glColor3f(1.0f,1.0f,0.0f);    // Color Yellow
    glVertex3f( 1.0f,-1.0f,-1.0f);    // Top Right Of The Quad (Back)
    glVertex3f(-1.0f,-1.0f,-1.0f);    // Top Left Of The Quad (Back)
    glVertex3f(-1.0f, 1.0f,-1.0f);    // Bottom Left Of The Quad (Back)
    glVertex3f( 1.0f, 1.0f,-1.0f);    // Bottom Right Of The Quad (Back)
    glColor3f(0.0f,0.0f,1.0f);    // Color Blue
    glVertex3f(-1.0f, 1.0f, 1.0f);    // Top Right Of The Quad (Left)
    glVertex3f(-1.0f, 1.0f,-1.0f);    // Top Left Of The Quad (Left)
    glVertex3f(-1.0f,-1.0f,-1.0f);    // Bottom Left Of The Quad (Left)
    glVertex3f(-1.0f,-1.0f, 1.0f);    // Bottom Right Of The Quad (Left)
    glColor3f(1.0f,0.0f,1.0f);    // Color Violet
    glVertex3f( 1.0f, 1.0f,-1.0f);    // Top Right Of The Quad (Right)
    glVertex3f( 1.0f, 1.0f, 1.0f);    // Top Left Of The Quad (Right)
    glVertex3f( 1.0f,-1.0f, 1.0f);    // Bottom Left Of The Quad (Right)
    glVertex3f( 1.0f,-1.0f,-1.0f);    // Bottom Right Of The Quad (Right)
  glEnd();            // End Drawing The Cube
glFlush();
}


void animation(void)
{

     yRotated += 0.01;
     xRotated += 0.02;
    DrawCube();
}


void reshape(int x, int y)
{
    if (y == 0 || x == 0) return;  //Nothing is visible then, so return
    //Set a new projection matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    //Angle of view:40 degrees
    //Near clipping plane distance: 0.5
    //Far clipping plane distance: 20.0

    gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y);  //Use the whole window for rendering
}

int main(int argc, char** argv){

glutInit(&argc, argv);
//we initizlilze the glut. functions
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowPosition(100, 100);
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(DrawCube);
```

```
glutReshapeFunc(reshape);
//Set the function for the animation.
glutIdleFunc(animation);
glutMainLoop();
return 0;
}
```

OUTPUT:



WINDMILL

//PROBLEM STATEMENT:-WRITE A PROGRAME IN OPENGL ON LINUX PERFORM TO ANIMATE ANY ONE SCENE (WIND MILL)

```
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>

int frameNumber = 0;

void drawWindmill()
{
        int i;
        glColor3f(1.0,1.0,0.0);
        glBegin(GL_POLYGON);

        glVertex2f(-0.05f, 0);
        glVertex2f(-0.05f, 3);
        glVertex2f(0.05f, 3);
        glVertex2f(0.05f, 0);

   glEnd();

        glTranslatef(0,3,0);
        glColor3f(1.0,0.0,0.0);
        glRotated(frameNumber * (180.0/45), 0, 0, 1);

                for (i = 0; i < 4; i++)
                {

                                glRotated(90, 0, 0, 1);
                                glBegin(GL_POLYGON);
                                glVertex2f(0,0);
                                glVertex2f(1.0f, 0.2f);
                                glVertex2f(1.0f,-0.2f);
                        glEnd();
           }
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glPushMatrix();
        glTranslated(2.2,1.6,0);
        glScaled(0.4,0.4,1);
        drawWindmill();
        glPopMatrix()


        glPushMatrix();

        glTranslated(3.7,0.8,0);

        glScaled(0.7,0.7,1);

        drawWindmill();

        glPopMatrix();



        glutSwapBuffers();
}



void doFrame(int v)
{
   frameNumber++;
   glutPostRedisplay();

   glutTimerFunc(10,doFrame,0);
}



void init()
{
        glClearColor(0,0,0,0);

        glMatrixMode(GL_PROJECTION);

        glLoadIdentity();

        glOrtho(0, 7, -1, 4, -1, 1);
```

```
            glMatrixMode(GL_MODELVIEW);
}



int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(700,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("WINDMILL");

    init();

    glutDisplayFunc(display);
    glutTimerFunc(200,doFrame,0);
    glutMainLoop();

    return 0;
}
```

OUTPUT: